# VelocityScriptingApi

## Java Scripting for Velocity

### Use cases

To be able to call Velocity engine from the standardized Java Scripting API (JSR 223).

### Coverage

Scripts are not Compilable. We rely on the internal Velocity cache for the parsing and re-use of templates.

No script generation method is to be considered, neither is any use of the Invokable interface.

Scripting of Java objects is already natural to Velocity, apart from Objects instantiation (currently possible only via an ugly hack).

(...)

### Terminology

A **script** corresponds to a Velocity template.

The **script context** corresponds to a Velocity running environment.

The **scripting engine** is Velocity and its corresponding discovery mechanism.

A **scripting engine instance** is a Velocity instance.

A **scripting engine instance metadata** corresponds to the Velocity instance configuration.

A **scripting scope** is a wrapper around a Velocity context.

Scripting scopes contain **bindings**, which correspond to Velocity contexts.

There can be **chained scripting scopes**, which corresponds to Velocity context chaining (to be developed - Global Scope & Engine Scope).

**Script context writers** correspond to Velocity writers.

**Script exceptions** are wrappers around Velocity exceptions (and can have filename and line number informations).

### Questions

- What does the scripting engine eval() method return? It could be the modified Velocity context resulting from the rendering (if so, do we make those contexts immutable?).
- May it be pertinent to reuse the Velocity Tools framework, rather than directly invoking Velocity engine objects? Maybe both approaches should be supported, as the former helps populating Velocity context with standard tools, and the latter allows less marshalling and configuration.
- Also check possible interactions with the Bean Scripting Framework.
- How shall we expose string templates evaluation?

### Design Choices

Threading: MULTITHREADED

language formal name = one of *velocity*, *vtl*, *velocity-template-language*

mime type = *text/x-velocity*

extension = *vhtml* (my preference) or *vtl*

### Diagram

Following will be the inter relationship between fundamental jsr 223 API classes.

ScriptEngine — ScriptContext — * Namespace

Namespace — Scope — * Attributes

Namespace — Reader

Namespace — Writer

ScriptEngineInfo — ScriptEngineFactory — * ScriptEngine

HttpScriptContext IS-A ScriptContext

Namespace IS-A java.util.Map